using XML technology while shielding the application programmer from many of the associated complexities.

The business services layer includes the core classes of an ApiService class, a Message class, a Field class and a BusinessService class. The application programmer deals with Message class and Field class while developing custom application code to integrate the business services layer into an e-commerce software architecture. ApiService class and BusinessService class transparently operate in cooperation with Message class and Field class to handle the details of XML as well as translation of information for the different delivery technologies within the end-user systems layer.

ApiService class together with BusinessService class, Message class and Field class handle the receipt of a request for data from the front-end systems layer and conversion of request parameters within the request to an input message. In addition, ApiService class directs execution of custom application code (contained in a subclasses of BusinessService) based on the request. The ApiService class also directs the creation of an output message from data provided by the custom application code in response to the request. Further, the ApiService class directs the translation of the output message to the desired format.

Message class and Field class act as wrappers to the functionality of a DOM (Document Object Model) class. Moreover, Message class and Field class include wrapping logic for an XSL Application Programming Interface (API). These wrappers limit the data structures that can be represented to "messages" containing "fields" of simple data types (string, long, integer, boolean) or group fields. Limiting the richness of the data structures to what is essential, while providing a generic translation mechanism, advantageously simplifies development of a business services application within the e-commerce software architecture.

An interesting feature of the business services layer is the ability of the application programmer to define both a long field name and a short field name for each field. The long or short field names may be selected as a mode of operation depending on the runtime environment. Where speed and/or bandwidth consumption is outweighed by the desire for longer more readily understood field names, the long field names may be selected; resulting in larger quantities of data being transferred

among the layers. Conversely, where minimizing the volume of data is a priority, the short names may be selected. Selection of the mode of operation is performed with a static variable called the mode debug flag.

Another interesting feature involves identification of the data types for the fields. Those fields that routinely appear in the input and output messages may be defined in a MESSAGEDEFINITION class within the business services layer. The data type for fields associated with a particular subclass of BusinessService class (a particular request), however, are defined within the custom application code of that particular subclass. As such, fields that are repetitively utilized within different subclasses of BusinessService class need only be defined once in the business services layer.

These and other features and advantages will become apparent upon a review of the following detailed description of the presently preferred embodiments of the invention viewed in conjunction with the appended drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of one embodiment of an e-commerce software architecture.

FIG. 2 is a more detailed block diagram of the e-commerce software architecture of FIG. 1.

FIG. 3 is a process flow diagram illustrating operation of the e-commerce software architecture illustrated in FIG. 2.

FIG. 4 is a second part of the process flow diagram of FIG. 3.

FIG. 5 is a third part of the process flow diagram of FIG. 3.

FIG. 6 is a fourth part of the process flow diagram of FIG.3.

FIG. 7 is a fifth part of the process flow diagram of FIG. 3.

FIG. 8 is a sixth part of the process flow diagram of FIG. 3.

FIG. 9 is a more detailed block diagram of an exemplary embodiment of a portion of the e-commerce software architecture of FIG. 2.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The presently preferred embodiments describe an extensible markup language (XML) based e-commerce software architecture forming a business services application. The architecture includes a business services layer that provides a framework for interfacing between information within a back-end systems layer and users operating a front-end systems layer. Users may request the information via the front-end systems layer using a plurality of different delivery technologies. The different delivery technologies operate with at least one of multiple presentation formats. The framework of the XML based e-commerce software architecture provides a flexible and generic approach to translating requests from the different delivery technologies. In addition, information resulting from the requests also utilizes the framework for translation of the information to a format compatible with the different delivery technologies. As such, the architecture provides a relatively simple configuration that is easier and cheaper to maintain than straight code.

FIG. 1 is a block diagram illustrating the layers of one embodiment of an e-commerce software architecture 10 that forms a business services application. The e-commerce software architecture 10 includes an end-user systems layer 12, a front-end systems layer 14, a business services layer 16 and a back-end systems layer 18. The various layers are illustrative designations to categorize software and/or hardware and corresponding functionality. Greater or fewer layers may be used to illustrated the e-commerce software architecture 10 in other embodiments. As used herein, the term "business services application(s)" may include any business-related software application providing access to data requested by users, such as, for example, financial services applications for home banking, brokering or electronic marketing.

The end-user systems layer 12 includes delivery technologies allowing a user to interface with the e-commerce software architecture 10. Exemplary delivery technologies include an Internet browser, a telephone, a wireline communication device, a wireless communication device, a wireless application protocol (WAP) device or any other software, hardware, or a combination thereof allowing a user to interface with the front-end systems layer 14. During operation, the end-user systems layer 12 of one embodiment provides for entry of user requests for data and access to information/data resulting from the user requests.